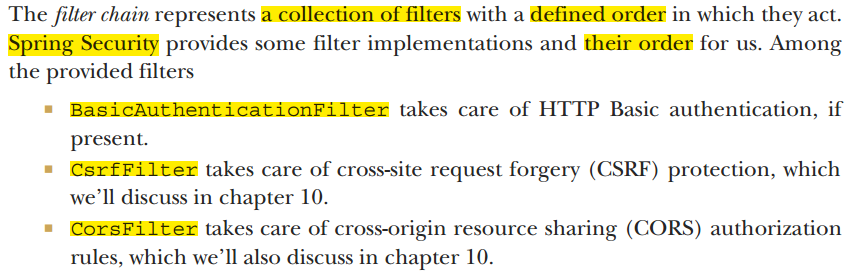
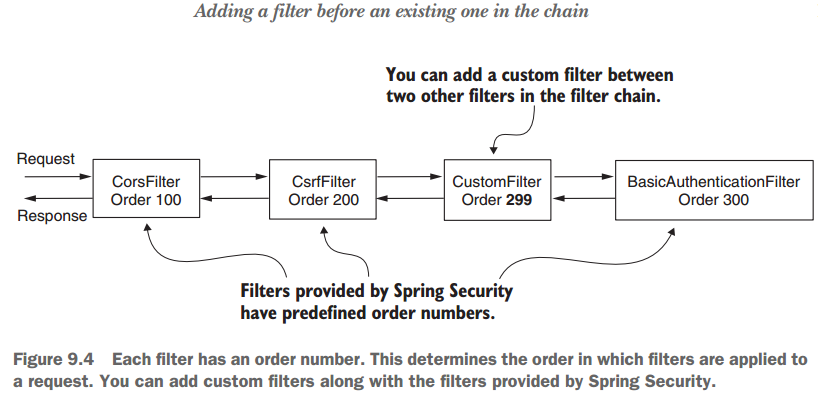
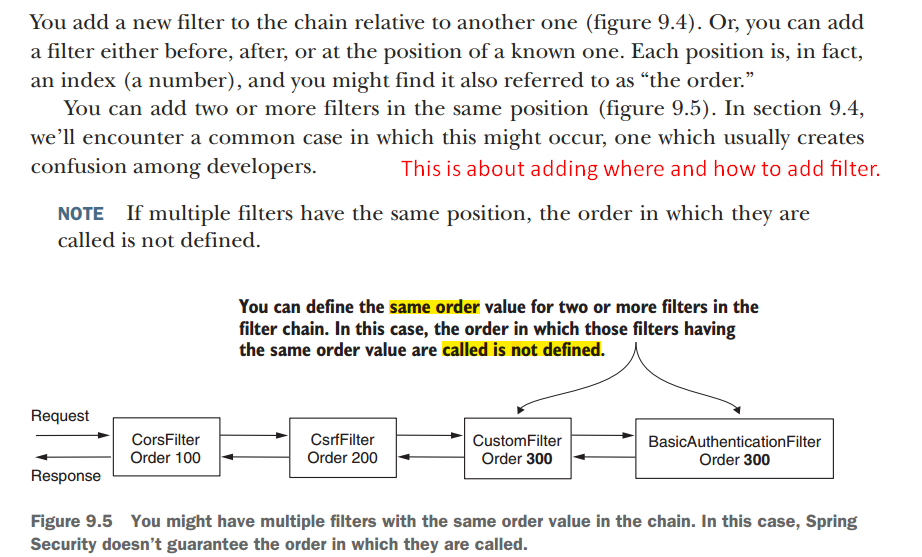
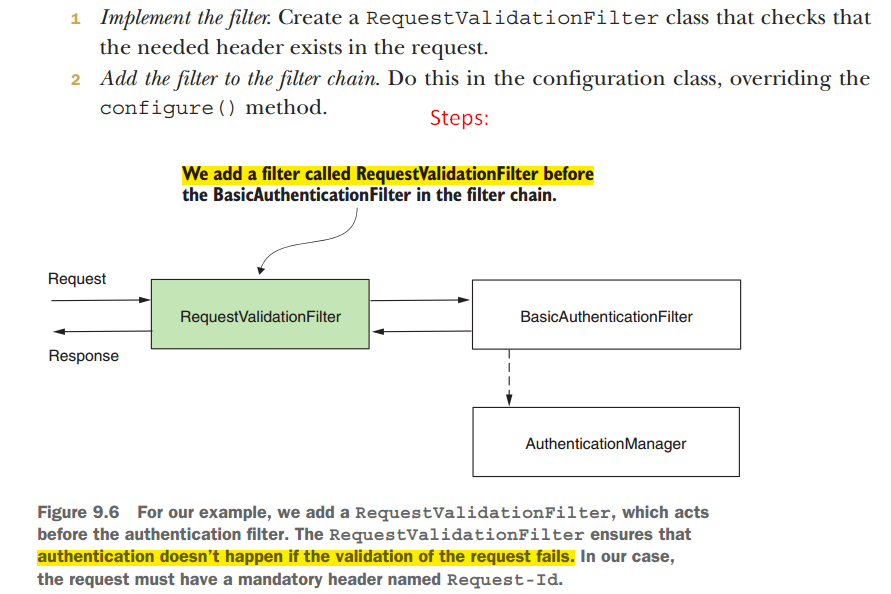
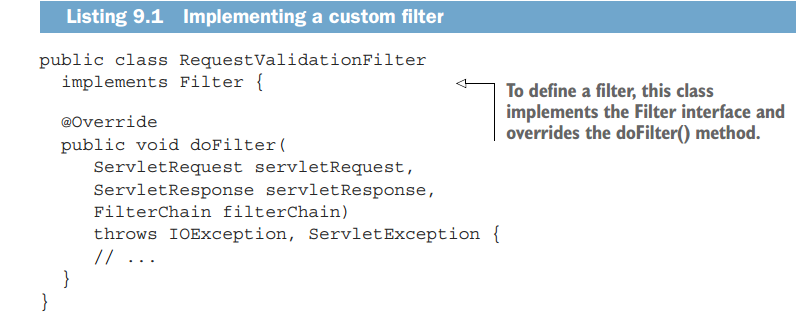
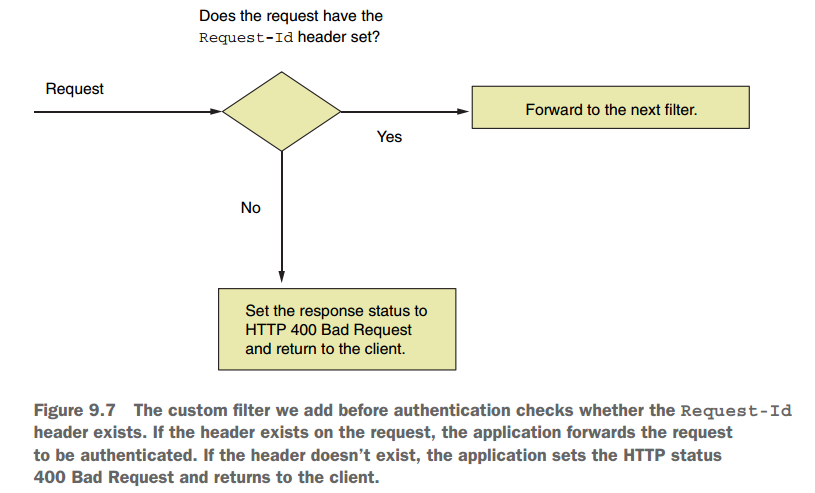
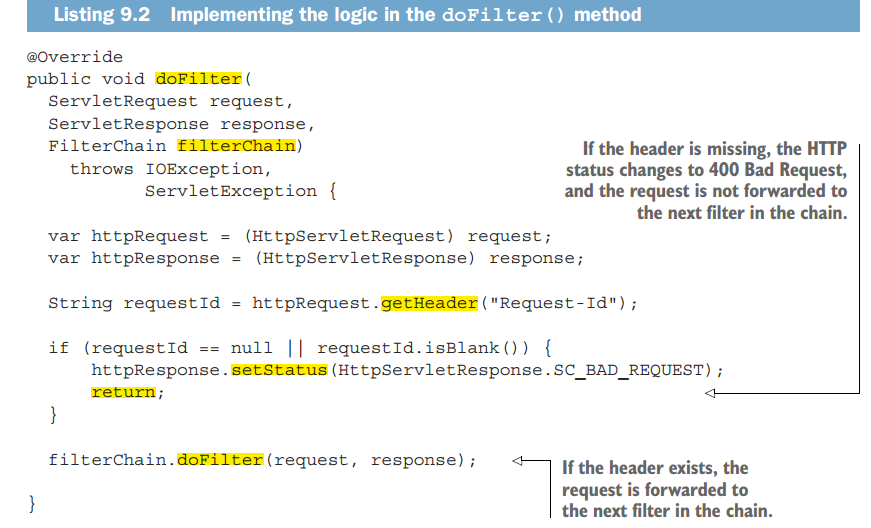
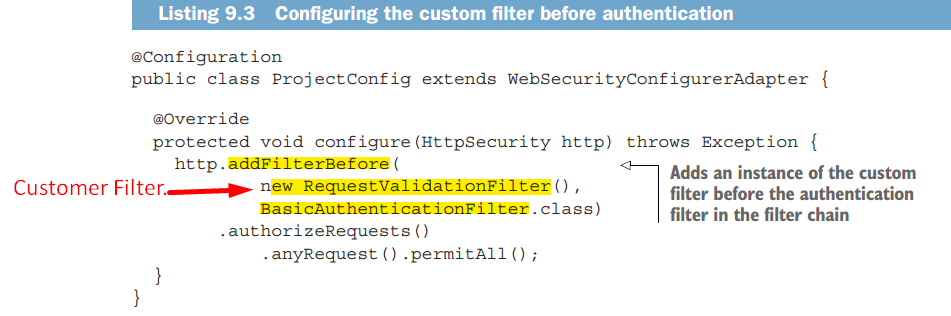
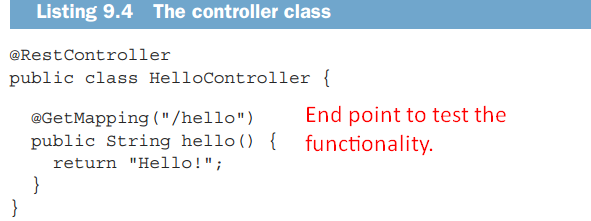
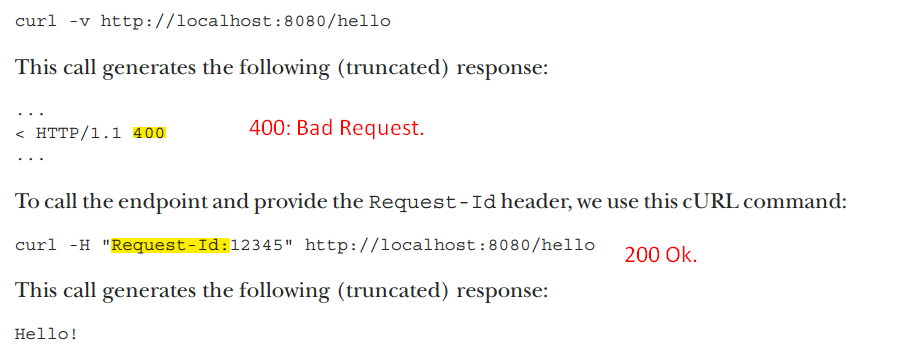
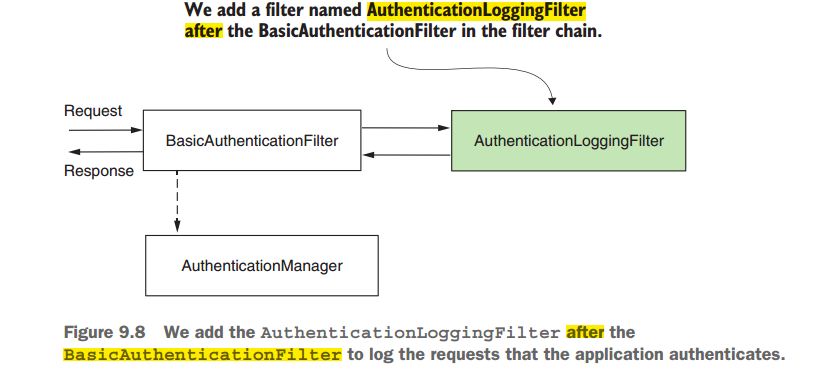
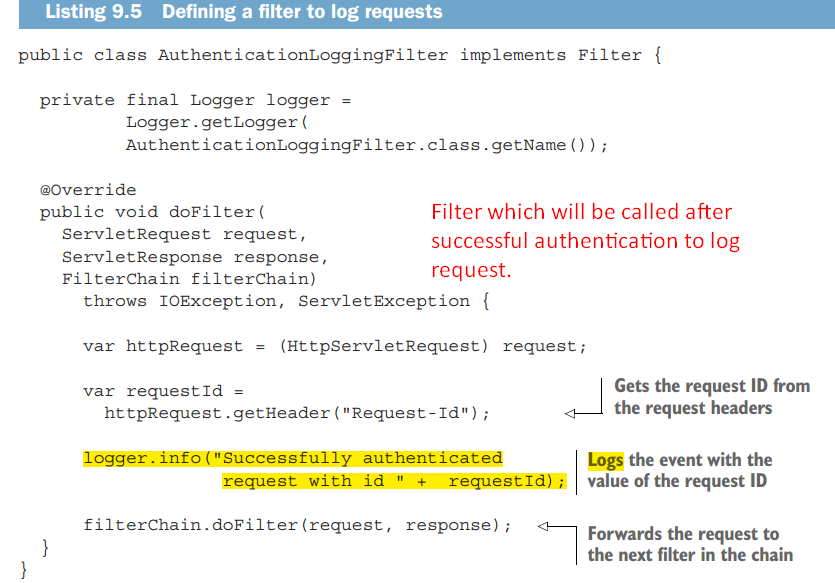
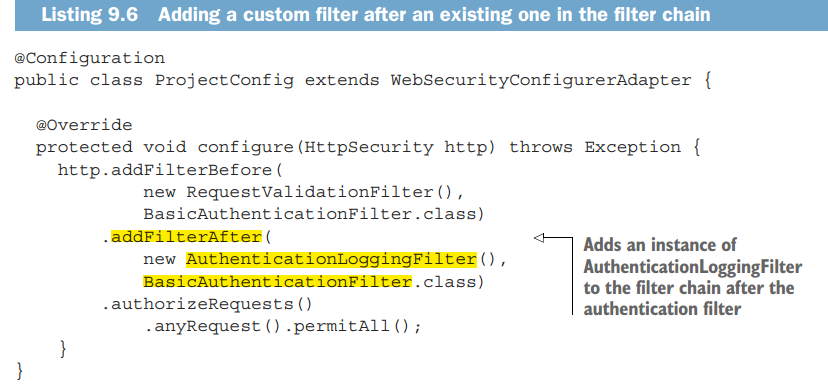
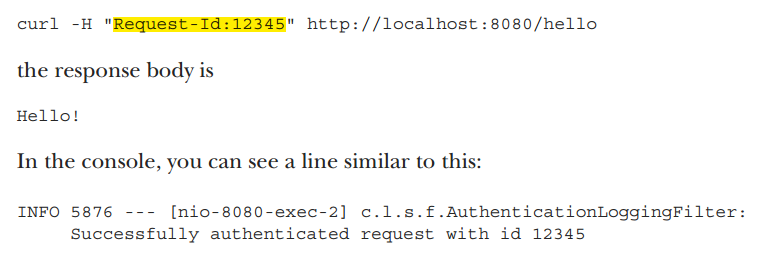
9.1 Implementing filters in Spring Security Architecture

1. **Agenda**:
   1. How filter and filter chain works?
2. We learnt in previous chapters that the authentication filter intercepts the request and delegates the authentication responsibility to authentication manager.
3. The filters in Spring Security are typically HTTP Filters.   
   We can create filters by implementing javax.servlet.Filter interface.  
   Override doFilter() to implement its logic.
4. 
5. See when you call HttpSecurity.httpBasic(), BasicAuthenticationFilter is added to the Spring Security Filter Chain.
6. So, depending on the configuration, filter chain gets affected.
7. 
8. 

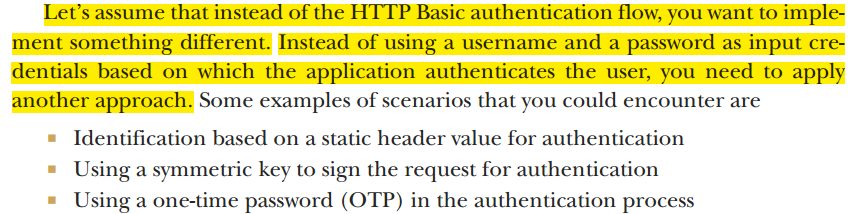
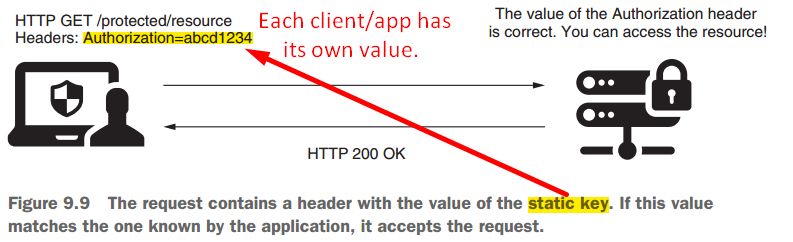
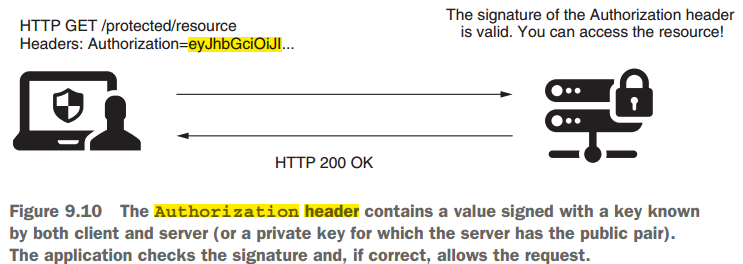
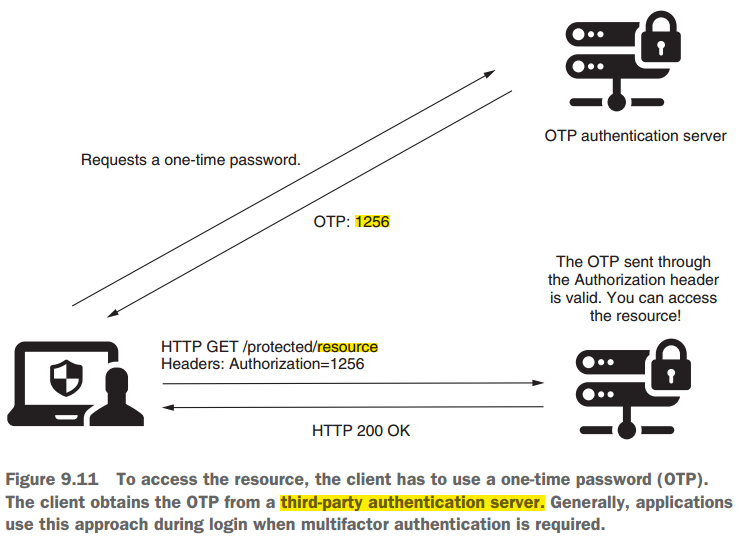
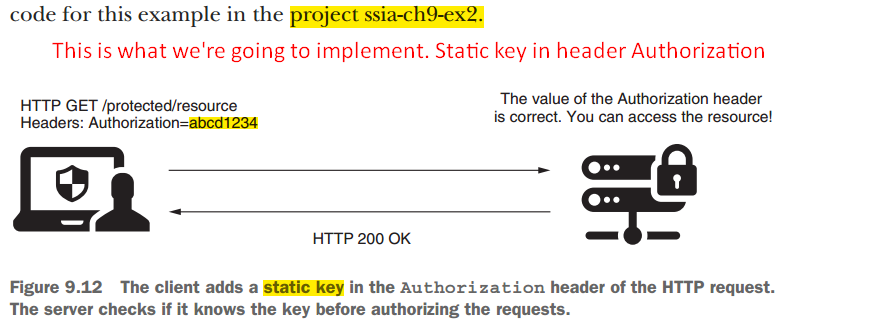
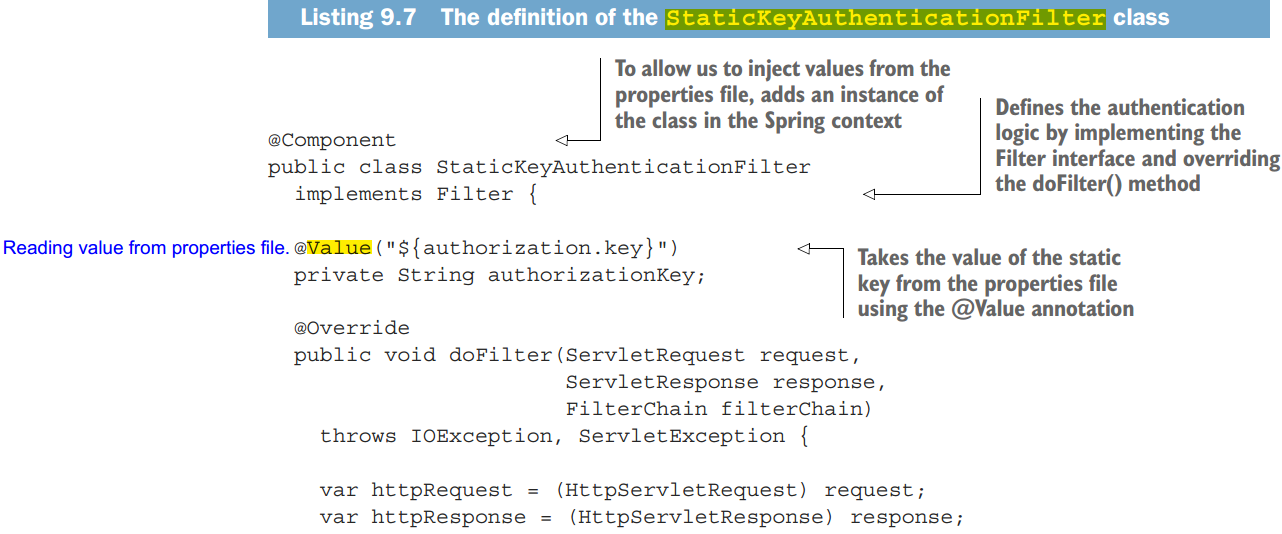
9.2 Adding a filter before an existing one in the chain

1. **Agenda**:
   1. How to add HTTP custom filters before an existing one in the filter chain.
2. **Requirement**:
   1. Implement filter checking for existence for Request-Id header and add it to Spring Security filter chain.
   2. Before application performs authentication, we want to check whether this header is present or not.
3. **See project** 🡺 ssia-ch9-ex1
4. 
5. 
6. 
7. 
8. 
9. 
10. **Two cases**:
    1. If Request-Id header is present then HTTP Status 200 Ok.
    2. Otherwise HTTP Status 400 Bad Request.
11. 

9.3 Adding a filter after an existing one in the chain

1. **Agenda**:
   1. Adding a filter after an existing one in the filter chain.
2. **Application:**
   1. When you want to execute some logic after something already existing in the filter chain.  
      Say, you have to execute some logic after the authentication process.  
      Examples for this could be notifying a different system after certain authentication events or logging and tracing purposes.
3. We will do
   1. Log all successful authentication events by adding a filter after the authentication filter.
4. 
5. 
6. 
7. 

9.4 Adding a filter at the location of another in the chain

1. **Agenda**:
   1. How to add a custom filter in place of another filter in the Spring Security filter chain
2. **Application**:
   1. You can use this approach when providing a **different implement** for a responsibility that is already assumed by one of the filters known by Spring Security.  
      A typical scenario is authentication.
3. 
4. **Identification based on a static header value for authentication**:  
   
   1. **Procedure**:
      1. The client sends a string to the app in the header of HTTP request, which is always the same.  
         **NOTE**: App stores all these values already in some DB or secrets vault.
      2. Based on this static value, the app identifies the client.
   2. **Disadvantage**: This approach offers weak security related to authentication
   3. **Application**: Architects and developers often choose it in calls b/w backend applications for its simplicity and its fast execution as its implementation doesn’t do complex calculations.
   4. This way, static keys used for authentication represent a compromise where developers rely more on the infrastructure level in terms of security and also don’t leave the endpoints wholly unprotected.
5. **Using a Symmetric Key to sign the request for authentication**:  
   
   1. In this scenario using symmetric key to sign and validate requests, both client and server know the value of a key (client and server share the key).
   2. **Procedure**:
      1. Client uses this key to sign a part of the request (for example, to sign the value of specific headers).
      2. The server checks if the signature is valid using the same secrete key.  
         **NOTE**: Server stores individual keys for each client in DB or secrets vault.  
         **NOTE**: Similarly, you can use a pair of asymmetric keys.
6. **Using a one-time password (OTP) in the authentication process:  
   **
   1. User receives OTP in the authentication process via a message or by using an authentication provider app like **Google Authenticator**.
7. Let’s implement an example to demonstrate how to apply a custom filter.
8. **Scenario**:
   1. To be authenticated, the user must add the correct value of the static key in the Authorization Header as presented in figure 9.12.
9. 
10. We will start with implementing the filter, named **StaticKeyAuthenticationFilter**.
11. 
12. 